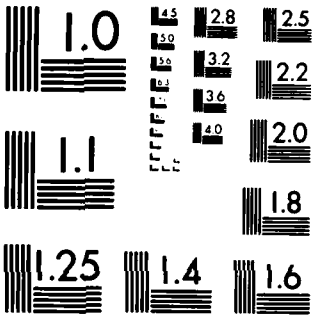
AD-A152 895 OPTIMAL PARALLEL ALGORITHMS FOR GRAPH CONNECTIVITY(U) 1/1
HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB
J H REIF 1984 TR-08-84 N00014-80-C-0647

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

OPTIMAL PARALLEL ALGORITHMS FOR GRAPH CONNECTIVITY

John H. Reif

TR-08-84

Harvard University

Center for Research in Computing Technology

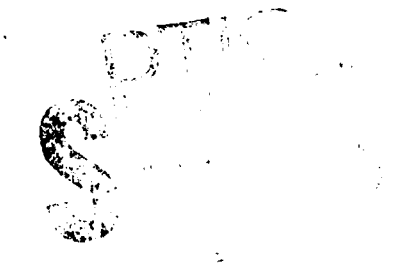
Aiken Computation Laboratory
33 Oxford Street
Cambridge, Massachusetts 02138

2

OPTIMAL PARALLEL ALGORITHMS FOR GRAPH CONNECTIVITY

John H. Reif

TR-08-84



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A152095 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OPTIMAL PARALLEL ALGORITHMS FOR GRAPH CONNECTIVITY	5. TYPE OF REPORT & PERIOD COVERED Technical Report
	6. PERFORMING ORG. REPORT NUMBER TR-08-84
7. AUTHOR(s) John H. Reif	8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0647
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harvard University Cambrdige, MA 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800 North Quincy Street Arlington, VA 22217	12. REPORT DATE 1984
	13. NUMBER OF PAGES 23
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) same as above	15. SECURITY CLASS. (of this report)
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

graph connectivity, parallel algorithms, optimal algorithms, randomized algorithms,

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

see reverse side.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

0. Abstract

We give a new randomized parallel RAM algorithm for finding a spanning forest of an undirected graph in logarithmic time. These time bounds hold with arbitrary high probability for any input graph (i.e., we do not assume random input; these bounds hold for the worst case input graph). This result assumes a parallel RAM model which allows both concurrent writes and concurrent reads.

Furthermore, we show that if the graph is not very sparse (i.e., if the number of edges is at least a logarithmic squared factor more than the number of vertices) than we can achieve a *linear processor · time product* (even for logarithmic time bounds) for finding a spanning tree--which is *optimal* for the parallel RAM model. Furthermore, we can also achieve a linear processor, time product for even sparser graphs with only slight time increase.

OPTIMAL PARALLEL ALGORITHMS FOR GRAPH CONNECTIVITY

John H. Reif

Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, Mass. 02138

0. Abstract

We give a new randomized parallel RAM algorithm for finding a spanning forest of an undirected graph in logarithmic time. These time bounds hold with arbitrary high probability for any input graph (i.e., we do not assume random input; these bounds hold for the worst case input graph). This result assumes a parallel RAM model which allows both concurrent writes and concurrent reads.

Furthermore, we show that if the graph is not very sparse (i.e., if the number of edges is at least a logarithmic squared factor more than the number of vertices) than we can achieve a *linear processor · time product* (even for logarithmic time bounds) for finding a spanning tree--which is *optimal* for the parallel RAM model. Furthermore, we can also achieve a linear processor, time product for even sparser graphs with only slight time increase.

1. Introduction

The performance of a parallel algorithm can be specified by bounds on its principle resources: time and processors. For most nontrivial graph problems, the product $P \cdot T$ of the number of processors P and the execution time T is lower bounded by $\Omega(n+m)$, where n,m are the number of vertices and edges, respectively of the input graph in adjacency list representation. Thus for these graph problems, an algorithm is *optimal* if $P \cdot T = O(n+m)$. Of course if we have an optimal algorithm with time T, then we also have (by the obvious processor simulation) optimal algorithms for any time bound T', where $T \leq T' \leq O(n+m)$.

The depth first search algorithm of [Tarjan, 72] was very successful in producing optimal sequential RAM algorithms for a large number of graph problems such as undirected graph connectivity and biconnectivity [Hopcroft and Tarjan, 73]. No optimal graph searching method has been proposed for parallel RAM, for polylog time bounds, except in the special case where the graph is extremely dense (i.e., $m = \Omega(n^2)$).

Previous work in parallel RAM algorithms has yielded few optimal results for graph connectivity. [Chin, Lam and Chen, 82] give $O(\log n)^2$ time connectivity algorithms requiring $(n^2+m)/(\log n)^2$ processors, which is optimal only if $m = \Omega(n^2)$. [Reif, 82a] gives a $\tilde{O}(\log n)^{\dagger}$ time connectivity algorithm for undirected graphs using randomization (see also [Reif, 82b] for a description of randomized P-RAMs), but requiring $n^2 m$ processors. All these algorithms assumed the parallel RAM model generally known as the P-RAM, (see

---

†Note: We use the notation $T(n) = \tilde{O}(f(n))$ if $\forall \alpha > 1 \; \exists c > 0$ such that $\text{Prob}(T(n) \leq c\, f(n)) \geq 1 - n^{-\alpha}$ for sufficiently large n.

[Fortune and Wyllie, 78]) where concurrent writes on the same memory cell are disallowed.

The WP-RAM (see [Shiloach and Vishkin, 81]) is a stronger parallel RAM model where concurrent writes on the same memory cell are allowed, and resolved arbitrarily. This machine model has the advantage that graph problems appear to be easier to solve on this model. The P-RAM can simulate the WP-RAM with only logarithmic factor of time increase and the same number of processors, by use of the parallel sorting algorithm of [Reischuk, 81].

In particular, [Shiloach and Vishkin, 82] give an $O(\log n)$ time, $O(n+m)$ processor WP-RAM algorithm for undirected graph connectivity and [Tarjan and Vishkin, 83] recently extended this result to biconnectivity. These algorithms have processor, time product $\Omega((m+n)\log n)$, which is a logarithmic factor more than optimal. [Vishkin, 84] gives an almost optimal n processor, and $O(\log n \log^* n)$ time bound for finding the number of successors on a linear list of length n. Vishkin conjectured that randomized techniques would also be needed to get optimal parallel graph connectivity algorithms.

1. Organization of our Paper and Statement of Results

In Section 2, we give our randomized WP-RAM algorithm for finding a spanning forest of a graph in simultaneously $\tilde{O}(\log n)$ time and $n+m$ processors. The randomized algorithm is surprisingly simple (in fact considerably simpler than previously known algorithms), and so seems to be useful in practice.

In Sections 3 and 4 we describe some modifications of our algorithms which results in an optimal processor·time product of $\tilde{O}(m+n)$. In particular

Section 3 gives an $\tilde{O}(\log n)$ time and $m/\log n$ processor algorithm for the case the number of edges m is at least $n(\log n)^2$. Furthermore, Section 4 gives an $\tilde{O}(\log n \log\log n)$ time and $m/(\log n \log\log n)$ processor algorithm for the case the number of edges m is at least $n \log n \log\log n$.

In Section 5 we give as an interesting application of these techniques an optimal parallel algorithm for finding the biconnected components of any undirected graph. This algorithm has the same complexity bounds as our parallel spanning forest algorithms. In the full paper we give applications to finding minimum spanning trees and Euler cycles.

2. A Randomized Parallel Algorithm for Spanning Forests

Let $G = (V,E)$ be an undirected graph with vertex set $V = \{1,\ldots,n\}$ and undirected edge set E of size m. Let its *directed edges* be $D(E) = \{(u,v) \mid \{u,v\} \in E\} \cup \{(v,u) \mid \{u,v\} \in E\}$. We shall associate a distinct processor with each vertex and edge of G. For each vertex $v \in V$, we shall have integer variables $R(v)$, $sex(v)$, and $link(v)$, and for each directed edge $(u,v) \in D(E)$ we have a boolean variable tree-edge(u,v). The following algorithm is to be executed by a WP-RAM as defined in Section 1. We assume that each processor $v \in V$ is provided with an independent random bit generator $RAND_v(0,1)$.

Algorithm RANDOM-MATE

Input graph $G = (V,E)$

Initialization:

```
begin
   for each v ∈ V in parallel do R(v) ← v
   for each (u,v) ∈ D(E) in parallel do tree-edge(u,v) ← false
end
```

```
Main Body:
  Until (R(u) = R(v)) for all (u,v) ∈ D(E) do
    begin
      for each v ∈ V in parallel such that R(v) = v do
        if RAND_v(0,1) = 1 then sex(v) ← female
                           else sex(v) ← male
      for each (u,v) ∈ D(E) in parallel do mate(u,v)
      for each v ∈ V in parallel do R(v) ← R(R(v))
    end
  output spanning forest F = {{u,v} ∈ E | tree-edge(u,v)}

We define:

Procedure mate(u,v):
  if sex(R(u)) = female and sex(R(v)) = male then
    begin
      attempt to assign link(R(u)) ← (u,v)
      if link(R(u)) = (u,v) then
        begin
          tree-edge(u,v) ← true
          R(R(u)) ← R(v)
        end
    end
```

Appendix I proves:

THEOREM 1. *The total number of parallel steps executed by* RANDOM-MATE *is* $\tilde{O}(\log n)$ (Note: see the footnote in the Introduction for the definition of our $\tilde{O}(\ )$ notation.)

3. An Optimal Algorithm for at Least $n(\log n)^2$ Edges

Let $G = (V,E)$ be an undirected graph with n vertices and $m \geq n(\log n)^2$ edges in adjacency list representation (we use a vector for each adjacency list).

Our modified algorithm RANDOM-MATE' will first reduce (with high likelihood) the number of edges and nodes to $m/\log n$. This is done by modifying RANDOM-MATE to execute its Main Body exactly $d_0\lceil \log n \rceil$ times, where d_0 is a constant. We shall assign a set P_v of $d_1\lceil \log n \rceil$ distinct processors for each vertex $v \in V$ where $d_1 \geq 2$ is a constant. Each processor $p \in P_v$ independently chooses a random list $E(p)$ of $d_0\lceil \log n \rceil$ directed edges in $D(E)$ (not necessarily distinct) departing vertex v.

In the Main Body of the resulting procedure RANDOM-MATE' is the same as that of RANDOM-MATE except that the statement

"for each $(u,v) \in D(E)$ in parallel do mate(u,v)"

is replaced with

"for each $v \in V$ and $p \in P_v$ in parallel do

begin
choose and delete the first edge e from $E(\neg)$
mate(e)
end"

After the $d_0\lceil \log n \rceil$ iterations of this modified Main Body, RANDOM-MATE' deletes each "loop" edge $(u,v) \in E$ such that $R(n) = R(v)$ using time $O(\log n)$ and $m/\log n$ processors. With high likelihood, the resulting graph is of size $m/\log n$. The final stage of RANDOM-MATE' is to apply the original algorithm RANDOM-MATE to this resulting graph, using $m/\log n$ processors. Appendix II proves:

THEOREM 2. *The total number of parallel steps executed by* RANDOM-MATE' *is* $\tilde{<}(\log n)$ *using* $d_1 m/\log n$ *processors.*

COROLLARY 2. *For all* T *such that* $\log n \leq T \leq n+m$, *we find a spanning tree in time* $\tilde{O}(T)$ *using an optimal number of processors* $P = m/T$ *assuming* $m \geq n(\log n)^2$.

4. An Optimal Parallel Algorithm for at Least n log n loglog n Edges

Let $G = (V,E)$ again be an undirected graph with $V = \{1,\ldots,n\}$ and $m = |E|$ edges which we assume are in adjacency list representation (when each adjacency list is given by a vector). To obtain optimal processor bounds for the case $m \geq n \log n \log\log n$, we assign a processor p to each $\lceil \log n \log\log n \rceil$ distinct consecutive directed edges in the adjacency list of each vertex $v \in V$. We name these processors by distinct numbers in $P = \{1,\ldots,\lceil (n+m)/(\log n \log\log n) \rceil\}$; so for each $p \in P$, $E(p)$ initially contains at most $\log n \log\log n$ edges departing the same vertex, and $D(E) = \cup_{p \in P} E(p)$.

Algorithm RANDOM-MATE"

Input graph $G = (V,E)$

Initialization

```
begin
  n ← |v|
  m ← |E|
  p ← {1,...⌈(m+n)/(log n loglog n)⌉}
  for each v ∈ V in parallel do R(v) ← v
  for each p ∈ P in parallel do
    begin
      construct E(p) as described above
      for each (u,v) ∈ E(p) do tree-edge(u,v) ← false
    end
end
```

```
Outer Loop:
  for  j = 1,...,⌈loglog n⌉  do
    begin
      Inner loop:
        for  i = 1,...,c1⌈log n⌉  do
          begin
            for each  v ∈ V  in parallel such that  R(v) = v  do
                if  RAND_v(0,1) = 1  then  sex(v) ← female
                                     else  sex(v) ← male
            for each  p ∈ P  in parallel do
                begin
                  choose a random  (u,v) ∈ E(p)
                  mate(u,v)
                end
            for each  v ∈ V  in parallel do  R(v) ← R(R(v))
          end
        for each  p ∈ P  in parallel
        do delete each edge  (u,v) ∈ E(p)  such that  R(u) = R(v)
    end
Execute until termination Main Body of RANDOM-MATE
Output  Spanning forest F = {{u,v} ∈ E | tree-edge(u,v)}
```

(Note the constants c_1, c_2 are chosen so as to achieve any given likelihood of success as bounded in Lemma 8.)

In Appendix III we prove:

THEOREM 3. RANDOM-MATE" *takes* $\tilde{O}(\log n \log\log n)$ *steps using* $(m+n)/(\log n \log\log n)$ *processors.*

COROLLARY 3. *For all* T *such that* $\log n \log\log n \leq T \leq n+m$, *we can find a spanning tree in time* $\tilde{O}(T)$ *using an optimal number of processors* $P = m/T$, *assuming* $m \geq n \log n \log\log n$.

5. Optimal Parallel Computation of Biconnected Components

Let $G = (V,E)$ be an undirected graph with $n = |V|$ vertices and $m = |E|$ edges. In Appendix IV we show

THEOREM 4. *The biconnected components of* G *can be computed in time* $\tilde{O}(T)$ *using* $P = m/T$ *processors, in the case either* ($T \geq \log n$ *and* $m \geq n(\log n)^2$) *or* ($T \geq \log n \log\log n$ *and* $m \geq n \log n \log\log n$).

The key idea is to reduce the problem (by repeated use of our optimal graph connectivity algorithms) to computing the biconnected components of a smaller graph G'' with only $O(n)$ edges and vertices.

REFERENCES

Angluin, D. and L.G. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Paths and Matchings," *J. Comp. Syst. Sci.* 18 (1979), pp. 155-193.

Awerbuch, B. and Y. Shiloach, "New Connectivity and MSF Algorithms for Ultracomputer and PRAM," IEEE Conf. on Parallel Comput., 1983.

Chin, F.Y., J. Lam and I. Chen, "Efficient Parallel Algorithms for Some Graph Problems", CACM, Vol. 25, No. 9 (Sept. 1982), p. 659.

Chernoff, H., "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Math. Statistics*, Vol. 23, 1952.

Fortune, S. and J. Wyllie, "Parallelism in Random Access Machines," Proc. 11th ACM Symp. on Theory of Computing, May, 1978.

Hirschberg, D.S., A.K. Chandra and D.V. Sarwate, "Computing Connected Components on Parallel Computers," CACM, Vol. 22 (1979), p. 461.

Hoeffding, W. "On the Distribution of the Number of Successes in Independent Trials," *Ann. of Math. Stat.* 27 (1956), 713-721.

Hopcroft, J.E. and R.E. Tarjan, "Efficient Algorithms for Graph Manipulation," *Comm. ACM* 16(6), 372-378 (1973).

Kucera, L., "Parallel Computation and Conflicts in Memory Acces," *Information Processing Letters*, Vol. 14, No. 2, April 1982.

Nath, D. and S.N. Maheshwari, "Parallel Algorithms for the Connected Components and Minimal Spanning Tree Problems," *Information Processing Letters*, Vol. 14, No. 2, April 1982.

Preparata, F.P. and J. Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation," CACM 24 (1981), 300-310.

Rabin, M.O., "Probabilistic Algorithms," in: *Algorithms and Complexity*, J.F. Traub (ed.), Academic Press, New York, 1976.

Reif, J., "Symmetric Complementation," 14th ACM Symp. on Theory of Computing, San Francisco, Cal., May 1982a, pp. 201-214.

Reif, J., "On the Power of Probabilistic Choice in Synchronous Parallel Computations," 9th Colloquium on Automata, Languages and Programming, Aarhus, Denmark, July 1982b, pp 442-456.

Reif, J.H., and P.G. Spirakis, "The Expected Time Complexity of Parallel Graph and Digraph Algorithms," TR-11-82, Aiken Computation Lab., Harvard Univ., Cambridge, Mass. (1982).

Reif, J.H. and L.G. Valiant, "A Logarithmic Time Sort for Linear Size Networks," Proc. Fifteenth Annual ACM Symp. on the Theory of Computing, pp. 10-16 (1983).

R. Reischuk, "A Fast Probabilistic Parallel Sorting Algorithm," Proc. of 22nd IEEE Symp. on Foundations of Computer Science (1981), 212-219.

Savage, C. and J. Ja'Ja', "Fast Efficient Parallel Algorithms for Some Graph Problems," *SIAM J. on Computing*, Vol. 10, No. 4 (Nov. 1981), p. 682.

Shiloach, Y. and U. Vishkin, "Finding the Maximum Merging and Sorting in a Parallel Computation Model," *J. of Algorithms*, Vol. 2, (1981), p. 88.

Shiloach, Y. and U. Vishkin, "An O(log n) Parallel Connectivity Algorithm," *J. of Algorithms*, Vol. 3 (1983), p. 57.

Tarjan, R.E., "Depth Forest Search and Linear Graph Algorithms," *SIAM J. Computing* 1(2), pp. 146-160 (1972).

Tarjan, R.E. and U. Vishkin, "An Efficient Parallel Biconnectivity Algorithm," Technical Report, Courant Institute, New York University, New York, 1983.

Ullman, J. "Computational Aspects of VLSI," Computer Science Press, 1983.

Vishkin, U., "Randomized Speed-Ups in Parallel Computation," *Proc. of the 16th Annual ACM Symp. on Theory of Computing*, Washington, D.C., April, 1984, 230-239.

APPENDIX 1. Proof of RANDOM-MATE

LEMMA 1. *On termination,* $F = \{\{u,v\} \in E \mid \text{tree-edge}(u,v)\}$ *is a spanning forest of* G.

Proof. We use an induction argument (easily seen to hold for the first iteration) that for just before iteration $i = 1,2,\ldots$ of the until loop,

(1) $F_i = \{\{u,v\} \in E \mid \text{tree-edge}(u,v)\}$ is a forest, and

(2) for all $u,v \in V$, $R(u) = R(v)$ iff u,v are in the same tree in F_i

(3) for all $v \in V$, $R(R(v)) = R(v)$.

Let T_w be the tree in F_i with root $w = R(w)$. (By the induction hypothesis, there is some such w). Suppose link(w) is assigned some edge (u,v). Then *sex*$(R(u)) \neq$ *sex*$(R(v))$ *and hence* $w = R(u) \neq R(v)$. This implies by the induction hypothesis that v is in a tree $T_{R(v)}$ of F_i distinct from T_w. Observe that $\{u,v\}$ is the only edge departing or entering T_w for which tree-edge(u,v) is assigned *true* on the i-th iteration. Hence F_{i+1} must be a forest. At the last step of the i-th iteration each vertex of T_w has $R(v)$ assigned $R(v)$, establishing the induction hypothesis for the state just before the $i+1$ iteration. □

A *maximal tree* spans an entire connected component of G.

Let n_i be the number of trees in F_i that are not of maximal size.

LEMMA 2. *If* $n_i \geq 1$, $\text{Prob}(n_{i+1} \leq (3/4)n_i) \geq 1/2$.

Proof. Let T_w be a tree of F_i which is not of maximal size. Then there exists at least one edge $(u,v) \in D(E)$ departing T_w to a distinct tree $T_{R(v)}$ where $R(u) = w \neq R(v)$. With probability 1/4, sex(w) = female and sex(R(v)) = male. Hence with probability at least 1/4, T_w is merged into some other tree (not necessarily $T_{R(v)}$) on the i-th iteration.

In the worst case, each nonmaximal size tree T_w in F_i has only one such departing edge (u,v) where $w \neq R(v)$, and each target v has a distinct $R(v)$. The random variable n_{i+1} is upper bounded in this worst case by a binomial variable $B_{n_i,3/4}$ (which is the sum of n_i independent Bernoulli variables, each with success probability $3/4$). Observe $B_{n_i,3/4}$ has mean $(3/4)n_i$ and n_{i+1} is upper bounded by $B_{n_i,3/4}$, so $\text{Prob}(n_{i+1} \leqslant (3/4)n_i) \geqslant \text{Prob}(B_{n_i,3/4} \leqslant (3/4)n_i) \geqslant 1/2$. □

Our time analysis will utilize the following probabilistic inequality which can be derived (see [Angluin and Valiant, 79]) from the bounds of [Chernoff, 52] and [Hoeffding, 56].

LEMMA 3. *Let* g *be the sum of* N *independent geometric variables. Let* μ *be the mean of* g. *Then for all* $\alpha > 0$ *there exists a* $c > 1$ *such that*

$$\text{Prob}(g \leqslant c\,\mu) < \frac{1}{2^{\alpha\mu}} \quad \textit{for } N \textit{ sufficiently large.}$$

Let $i_0 = 1$ and inductively for $k = 1,2,\ldots$ let i_k be the minimum number such that $n_{i_k} \leqslant (3/4)n_{i_{k-1}}$. By Lemma 2, each i_k is upper bounded by an independent geometric variable with expectation 2. Since $n_{i_{k_0}} < 1$ for $k_0 = \log_{4/3} n$, $I = \Sigma_{k=1}^{k_0} i_k$ is the total number of iterations of the until loop executed until termination. The expectation of I is $O(\log n)$. Each execution of this loop takes only a constant number of steps. Hence by Lemma 3, we have Theorem 1. □

APPENDIX II. Proof of RANDOM-MATE'

It is easy to show that Lemma 1 holds and further more its induction hypothesis also holds. Let $F_i' = \{(u,v) \in E | \text{tree-edge}(u,v)\}$ be the forest defined on the i-th iteration of the modified Main Body of RANDOM-MATE'. Let a i-*loop* be on edge which on the i-th iteration departs and enters the same tree of F_i'. Let a tree T_w of F_i' be *semi-active* if at least $1/\log n$ of the set of edges departing T_w are not i-loops. Let n_i' be the number of semi-active trees of F_i'.

LEMMA 5. *If* $n_i \geq 1$, $\text{Prob}(n_{i+1}' \leq (1-1/4e)n_i') \geq 1/2$.

Proof. Let T_w be a semiactive tree of F_i' which is not of maximal size with root $w = r(w)$. By the pigeon-hole principal, there are at least one vertex v of T_w where $R(v) = w$ and at least $1/\log n$ of its departing edges are not i-loops. Let $(u,v_1),\ldots,(u,v_{d_1 \log n})$ be the edges chosen by the $d_1 \log n$ processors of P_v on the i-th iteration. For each $j = 1,\ldots,d_1 \log n$, with independent probability at least $1/\log n$, $R(u) \neq R(v_j)$. (Note: the independence is due to the fact that the processors initially choose random edges independently of each other.) Furthermore, if $R(u) \neq R(v_j)$ then, $\text{Prob}(\text{sex}(R(u)) = \underline{\text{female}}$ and $\text{sex}(R(v_j)) = \underline{\text{male}}) = 1/4$. Since $d_1 \geq 2$, the probability that T_w does not merge into some other tree on the i-th iteration is at most $(1/4)(1-1/\log n)^{\log n} \leq (1/4)e^{-1}$. An argument similar to Lemma 2 then shows that n_{i+1}' is upper bounded by a binomial with expectation $(1-1/4e)n_i'$ and hence $\text{Prob}(n_{i+1}' \leq (1-1/4e)n_i')) \geq 1/2$. □

Recall that $d_0 \log n$ is the number of iterations of the Main Body of RANDOM-MATE'.

LEMMA 6. $\forall \alpha > 0\ \exists d_0 > 1$ *such that there are no semiactive trees in* $F'_{d_0\lceil \log n \rceil}$ *with probability of least* $1 - 1/n^{\alpha}$.

Proof. Let $i_0 = 0$ and inductively for $k = 1,2,\ldots$ let i_k be the minimum number such that $n'_{i_k} \leq (1-1/4e)n'_{i_{k-1}}$. By Lemma 5, each is bounded by an independent geometric variable with expectation 2. Let k_1 be the maximum number such that $\sum_{k=1}^{k_1} i_k \leq d_1\lceil \log n \rceil$. Now suppose there is an active tree in $F'_{d_0\lceil \log n \rceil}$ with probability more than $1/n^{\alpha}$. Then the bound on $\sum_{k=1}^{k_1} i_k$ given by Lemma 3 is violated, a contradiction. □

Since each iteration of the Main Body of RANDOM-MATE' takes only constant time, Lemma 6 implies that with probability at least $1-1/n^{\alpha}$, the number of the "nonloop edges" after execution of Main Body is at most $m/\log n$. By Theorem 1, the final execution of RANDOM-MATE on this reduced graph takes $\tilde{O}(\log n)$ time using $m/\log n$ processors. Hence we have proved Theorem 2. □

APPENDIX III. Proof of RANDOM-MATE"

Again, it is easy to show by induction that Lemma 1 holds

LEMMA 4. *On termination,* $F = \{\{u,v\} \in E | \text{tree-edge}(u,v)\}$ *is a spanning tree of* G.

Furthermore, the induction hypothesis stated in the proof of Lemma 1, also holds for RANDOM-MATE".

On the j-th iteration of the Outer Loop and the i-th iteration of the Inner Loop,

(1) let $F_{i,j} = \{\{u,v\} \in E | \text{tree-edge}(u,v)\}$

(2) let $E_{i,j} \subseteq D(E)$ be the set of edges remaining

(3) let $D_{i,j}(T_w) \subseteq E_{i,j}$ be the set of edges departing vertices in T_w, where T_w is a tree of $F_{i,j}$

(4) let a tree T_w of $F_{i,j}$ be *active* if at least half of the edges of $D_{i,j}(T_w)$ do not enter a vertex of T_w

(5) let $n_{i,j}$ be the number of active trees in $F_{i,j}$ that are not of maximal size.

The following Lemma is similar to Lemma 5.

LEMMA 7. *If* $n_{i,j} \geq 1$, $\text{Prob}(n_{i+1,j} \leq (7/8)n_{i,j}) \geq 1/2$.

Proof. Let T_w be an active tree of $F_{i,j}$ which is not of maximal size, and containing vertex $w = r(w)$. Then, since T_w is not of maximal size, there exists at least one edge $(u,v) \in D_{i,j}(T_w)$ entering a distinct tree $T_{R(v)}$ where $w \neq R(v)$. Furthermore, since T_w is assumed to be active, by the pigeon-hole principal for some $p \in P$ such that $E(p) \subseteq D_{i,j}(T_w)$ at least one half of the edges $(u,v) \in E(p)$ have $R(v) \neq R(u)$. Let (u,v) be any edge of $E(p)$. With probability 1/4, sex(w) = female and

sex(R(v)) = male when $R(u) = w$ and $R(v) \neq w$. Furthermore, if (u,v) is a random edge of $E(p)$, $\text{Prob}(R(v) \neq R(u)) \geq 1/2$. Hence with probability of least $1/8$, T_w is merged into some other tree on the i,j iteration. As in the proof of Lemma 1, we observe that the worst case is where there is exactly one edge $(u,v) \in D_{i,j}(T_w)$ such that $R(v) \neq w$, and when each $R(v)$ is distinct. In this worst case, $n_{i+1,j}$ is upper bounded by a binomial variable B which is the sum of $n_{i,j}$ independent Bernoulli variables, each with success probability $7/8$. But since the mean of B is $(7/8)n_{i,j}$ and B upper bounds $n_{i+1,j}$, $\text{Prob}(n_{i+1,j} \leq (7/8)n_{i,j}) \geq \text{Prob}(B \leq (7/8)n_{i,j}) \geq 1/2$. □

Recall that $c_1'\log n'$ is the number of iterations in Inner Loop.

LEMMA 8. $\forall \alpha > 0 \ \exists c_1 > 1$ *such that there are no active trees in* $F_{c_1'\log n'+1,j}$ *with probability at least* $1 - 1/n^{\alpha}$.

Proof. Let $i_{o,j} = 0$ and inductively for $k = 1,2,\ldots$ let $i_{k,j}$ be the minimum number such that $n_{i_{k,j}} \leq (7/8)n_{i_{k-1,j}}$. By Lemma 7, each $i_{k,j}$ is bounded by an independent geometric variable with expectation 2. Let k_2 be the maximum number such that $\Sigma_{k=1}^{k_2} i_{k,j} \leq c_1'\log n'$. Now suppose there is an active tree in $F_{c_1'\log n'+1,j}$ with probability more than $1/n^{\alpha}$. Then the bound on $\Sigma_{k=1}^{k_2} i_{k,j}$ given by Lemma 3 is violated, a contradiction. □

Since each iteration of the Inner loop takes only constant time, each iteration of the Outer loop takes total time $O(\max_{p \in P} |E(p)| + \log n)$. Lemma 8 implies that with probability at least $1 - 1/n^{\alpha}$, the number of edges assigned to each $E(p)$ decreases by at least a factor of two on each iteration

of the Outer loop. Furthermore, the number of vertices also decrease by at least a factor of two on each iteration of the Outer loop. Thus the total execution time of the 'loglog n' iterations of the Outer loop is

$$O(\max_{p \in P} |E(p)|(1 + 1/2 + \ldots) + \text{loglog } n \log n) \leq O(\log n \text{ loglog } n)$$

with probability at least $1 - 1/n^{\alpha-1}$, and hence is $\tilde{O}(\log n \text{ loglog } n)$.

After completing all these iterations, the size of the graph has decreased to $(m+n)/\log n$ with probability $1 - 1/n^{\alpha-1}$. We then can apply Theorem 1 to bound the execution time of the call to RANDOM-MATE (using $(m+n)/(\log n \text{ loglog } n)$ processors) to be $\tilde{O}(\log n \text{ loglog } n)$. Thus the total execution time of RANDOM-MATE" is $\tilde{O}(\log n \text{ loglog } n)$ using $(n+m)/(\log n \text{ loglog } n)$ processors, proving Theorem 3.

APPENDIX IV. Finding Biconnected Components

Algorithm BICONNECT

input undirected graph $G = (V,E)$

begin

[1] compute a spanning forest F of G

[2] Root each tree in F and compute its preordering and the number of tree descendants of each vertex.

[3] Construct a forest F' derived from F by adding a new induced vertex v_e for each edge $e \in F$ and in place of edge $e = \{u_1,u_2\}$ substitute edges $\{u_1,v_e\}$ and $\{v_e,u_2\}$.

[4] Construct a graph $G' = (V',E')$
where $V' = \{v_e | e \in F\}$ is the set of induced vertices
and $E' = \{f(u_1,u_2) | \{u_1,u_2\} \in E-F\}$
and $f(u_1,u_2) = \{v_{e_1},v_{e_2}\}$ where
if u_1, u_2 are unrelated in F (i.e., one is not the ancestor of the other) then e_1, $e_2 \in F$ are the tree edges entering u_1,u_2 from their parents, or if (without loss of generality) u_1 is the ancestor of u_2 in F, then $e_1 \in F$ is the tree edge departing u_1 on the tree path from u_1 to u_2 and $e_2 \in F$ is as previously described.

[5] Compute the set $C(G')$ of connected components of G'.

[6] Construct graph $G'' = (V \cup C(G'),E'')$ from F' by collapsing together all vertices in V' which are the same connected component of G'.

[7] Compute the set $B(G'')$ of biconnected components of G''.

[8] Merge together all biconnected components of G'' connected by articulation points in $C(G')$. (To do this, we construct a graph $G''' = (B(G''), E''')$ whose vertices are the biconnected components of G'' and each edge $\{B_1, B_2\} \in E'''$ connects biconnected components $B_1, B_2 \in B(G'')$ with a common articulation point in $C(G')$. Then we compute the connected components of G'''.)

[9] For each edge $e \in E$, let v_e be its induced vertex in V', let C_e be the connected component in $C(G')$ containing v_e, let S_e be the set of biconnected components in $B(G'')$ with articulation point C_e, let B_e be the connected component of G''' containing S_e.

Output B_e for each $e \in E$.

LEMMA 7. $\forall e_1, e_2 \in E$, e_1, e_2 *are in the same biconnected component of* G *iff* $B_{e_1} = B_{e_2}$.

Proof. If $C_{e_1} = C_{e_2}$, then we can find a path of tree edges in F from e_1 to e_2, and also a disjoint path of nontree edges in $E-F$ from e_1 to e_2, and hence e_1, e_2 are in the same biconnected component of G (however, the reverse is not necessarily true).

Suppose $S_{e_1} = S_{e_2}$ but $C_{e_1} \neq C_{e_2}$. Any biconnected component $B \in S_{e_1}$ is thus connected in G'' to both articulation points C_{e_1} and C_{e_2}. Using the fact that B is biconnected, we can find two disjoint paths p_1, p_2 in G between edges e_1', e_2' such that $C_{e_1'} = C_{e_1}$ and $C_{e_2'} = C_{e_2}$. But since $C_{e_1'} = C_{e_1}$, we can find disjoint paths p_1', p_2' in G from e_1 to e_1' and since $C_{e_2'} = C_{e_2}$, we can find disjoint paths p_1'', p_2'' in G from e_2' to e_2. Moreover, p_1', p_1, p_1'' can be shown to be disjoint from p_2', p_2, p_2''. Hence $p_1' \cdot p_1 \cdot p_1''$ and $p_2' \cdot p_2 \cdot p_2''$ are disjoint paths in G

from e_1 and e_2 so e_1, e_2 are in the same biconnected component (again, the reverse may not be true).

Suppose $B_{e_1} = B_{e_2}$ but $S_{e_1} \neq S_{e_2}$. By induction on the minimum length of paths in G"' from S_{e_1} to S_{e_2}, we can similarly construct two disjoint paths in G from e_1 to e_2. Hence e_1, e_2 are in the same biconnected component.

On the other hand, suppose e, e' are in the same biconnected component. Then there is a simple cycle C of G containing both e and e'. C can be written as the mod-two sum of some k basis cycles $C_1,\ldots,C_k$ with respect to the tree of F which is the spanning tree of the connected component of G containing e_1 and e_2. These can be ordered so that for each $i > 1$, C_i has at least one edge, say e_i, in common with C_j from $j < i$. It is easy to verify that $B_e = B_{e'}$ if e, e' are on the same basis graph. Suppose they are on the distinct basic cycles.

We assume an induction hypothesis that $B_{e_1} = \ldots = B_{e_{i+1}}$ for some i, $1 < i \leq k$. Then since e_i is in both C_{e_i} and C_{e_j} for some $j < i$, we can show $B_{e_j} = B_{e_i}$. Thus we have $B_{e_1} = \ldots = B_{e_k}$. But $e \in C_i$ and $e' \in C_j$ for some i and j. Hence $B_e = B_{e_i} = B_{e_j} = B_{e'}$. □

The spanning forest F computed in step [1], and the connected components computed in steps [5] and [8] can all be computed using our randomized algorithms RANDOM-MATE' or RANDOM-MATE". Since F and G" have each only O(n) edges and vertices, steps [2] and [7] can be computed in time O(log n) time with n processors by the results of [Tarjan and Vishkin, 83]. Note that the ancestor tests required in step [4] can be done (see [Tarjan, 72]) in constant time using the preordering and descendant numbering computed in step [2]. The graph constructions in steps [3], [4] and [8] can easily be done with a processor·time product O(m+n). Theorem 4 follows from Theorems 2 and 3. □

END

FILMED

5-85

DTIC